Context-Dependent Composition of Linear Operators

Abstract

A composition of fixed linear mappings is itself a single linear mapping. This simple fact often leads to the impression that chains of linear operators are fundamentally uninteresting as learning systems. This note argues that the situation changes completely once the individual operators become **context-dependent**. Rather than applying the same linear mapping to every input, each stage selects one mapping from a learned collection according to a context computed from the input. The resulting system remains locally linear while becoming globally piecewise linear through combinatorial composition. This note presents the construction without reference to any particular neural network architecture or routing mechanism and suggests that it may provide an interesting design space for machine learning and an interesting mathematical object in its own right.

---

1. Introduction

Consider a sequence of linear mappings

$$y = A_n A_{n-1} \cdots A_2 A_1 x.$$

Since matrix multiplication is associative, the entire chain is equivalent to a single matrix

$$y = Mx.$$

As a result, simply stacking fixed linear mappings does not create a richer family of functions. The individual matrices may have useful parameterizations or constraints, but the resulting mapping remains linear.

This observation naturally raises a question.

What is the smallest modification that turns a chain of linear operators into a genuinely nonlinear learning system while preserving the simplicity of linear computation?

The proposal explored here is that the linear mappings themselves remain entirely ordinary. The only additional ingredient is that the mapping chosen at each stage is allowed to depend on the input.

---

2. A Library of Linear Operators

Suppose each stage contains not one matrix but a small collection

$$\mathcal{A}^{(k)} = \left\{ A_1^{(k)}, A_2^{(k)}, \ldots, A_{m_k}^{(k)} \right\}.$$

A context function

$$c(x)$$

selects one operator from each library.

The resulting computation becomes

$$y = A^{(n)}_{c_n(x)} A^{(n-1)}_{c_{n-1}(x)} \cdots A^{(1)}_{c_1(x)} x.$$

For any particular input, the computation is simply a sequence of ordinary matrix multiplications.

Across different inputs, however, different operator sequences are selected.

---

3. Combinatorial Composition

The expressive power arises from composition rather than from any individual matrix.

As a simple example, suppose three stages each contain two matrices.

Stage 1:

$$A_0,\ A_1$$

Stage 2:

$$B_0,\ B_1$$

Stage 3:

$$C_0,\ C_1.$$

The possible computations are

$$C_i B_j A_k, \qquad i, j, k \in \{0, 1\}.$$

Although only six matrices are stored, eight different composite mappings become available.

In general,

- m choices at each stage,
- d stages,

produce

$$m^d$$

possible operator compositions while storing only

$$md$$

individual matrices.

Many compositions may coincide algebraically, but the combinatorial growth of possible computational paths is immediate.

---

4. Factorization of Computation

An important consequence is that the learned computation becomes naturally factorized.

Rather than learning one large mapping for every possible situation, the model learns reusable operator components.

A matrix selected early in the chain may participate in many different composite mappings.

Likewise, a matrix appearing near the output may combine with many different earlier choices.

Learning therefore acts on a collection of reusable computational factors rather than on independent monolithic mappings.

This reuse is not imposed by hand. It emerges from the compositional structure itself.

---

5. Learning

No particular learning algorithm is assumed.

If the operator libraries are differentiable parameters and the context selection mechanism permits gradient-based optimization, standard backpropagation can be used to update the selected matrices.

Viewed from this perspective, backpropagation learns a collection of reusable linear operators whose compositions solve different regions of the input space.

The role of learning is therefore twofold:

- adjust the individual linear operators,
- organize those operators into useful compositions through the context mechanism.

No claim is made here regarding the optimal routing mechanism or training procedure. The central observation is independent of these implementation choices.

---

6. Context

The context function is intentionally left unspecified.

It may be produced by

- binary tests,
- locality-sensitive hashing,

- learned routing,
- decision trees,
- attention,
- another neural network,
- or any mechanism capable of selecting operators.

The proposal does not depend on how context is computed.

Only the following property is required:

Different inputs may produce different compositions of linear operators.

---

7. A Different View of Nonlinearity

In many machine learning systems, nonlinearity is viewed primarily as an operation applied to intermediate activations.

An alternative viewpoint is that nonlinearity arises because different inputs follow different computational paths.

Under this interpretation, the individual computations remain linear.

The nonlinear behavior emerges from the input-dependent selection of those computations.

This separates two distinct concepts:

- linear computation,
- computational selection.

The latter determines which linear computation is performed.

---

8. Mathematical Questions

Independently of any practical application, the construction suggests several mathematical questions.

- How many distinct composite mappings can be generated from a given operator library?

- Under what conditions do different compositions become identical?

- How should operator libraries be parameterized to maximize expressive power?

- What classes of functions can be represented efficiently through context-dependent composition?

- Can useful computational factorizations emerge automatically during learning?

These questions combine ideas from linear algebra, semigroup theory, combinatorics, and machine learning.

---

9. Conclusion

The central idea of this note is deliberately modest.

A chain of fixed linear mappings is simply another linear mapping.

A chain of **context-selected** linear mappings is no longer fixed.

Its expressive power comes not from the complexity of the individual operators but from the combinatorial space of possible compositions.

This perspective does not prescribe a particular architecture, routing algorithm, or learning rule. Instead, it identifies a simple construction that appears to define a broad family of learning systems built from ordinary linear operators and input-dependent composition.

Whether this viewpoint leads to practical architectures, useful theory, or simply interesting mathematics remains an open question. The hope is that isolating the underlying principle makes that question easier to investigate.